Transcrição

Veremos uma última maneira de construirmos as tabelas da Distribuição de Frequências quando não temos classes previamente organizadas, como e

Agora, aprenderemos a **Regra de Sturges** que otimiza a escolha da quantidade de classes que teremos nas tabelas de distribuições, considerando son

$k = 1 + (10/3 \log_{10} n)$

Na parte "2.3 Distribuição de frequências para variáveis quantitativas (classes de amplitude fixa)", o primeiro passo é importar a biblioteca de análise com álgebra linear ou matrizes por exemplo.

A usaremos unicamente para podermos usar a função `log10` da regra. Então a importaremos como `np`.

Esta somente precisará do **número de observações** como o "n" da fórmula. Para o encontrarmos na distribuição do dataset, criaremos a variável `n`

Em seguida, executaremos a célula com `n`.

```
n= dados.shape
n
```

O resultado será o número de registros e a quantidade de variáveis: `(76840, 7)`.

Se quisermos apenas as observações, colocaremos `0` para `.shape[]` pegar somente o primeiro elemento.

```
n= dados.shape[0]
n
```

Agora que sabemos que `n` é igual a `76840`, descobriremos o número de classes de **amplitude fixa** por meio da regra de Sturges. Multiplicaremos p

```
k = 1 + (10 /3) * np.log10(n)
```

Executando a célula, veremos que o valor de `k` é igual a `17.285291187298853`.

Com isso, saberemos que uma boa forma de visualizarmos a tabela com a quantidade de registros que temos é de aproximadamente 17 classes.

Para arredondarmos o valor de fato, escreveremos que `k` é igual a `k.round()` recebendo `0` casas decimais na célula seguinte, e em seguida chama

```
k = k.round(0)
k
```

O retorno será de `17.0`, mas ainda queremos exibir um valor inteiro.

Logo, englobaremos `k.round(0)` na função `int()` do Python, resultando em `17` como queríamos.

```
k = int(k.round(0))
k
```

Anotaremos esta fórmula para criarmos o `k` sempre que precisarmos de seu valor.

O segundo passo é a mesma coisa que fizemos nas etapas anteriores com `value_counts()` e `cut()` .

O parâmetro `x` será a `Renda` também, o segundo `bins` será o número `17` de classes de mesma amplitude que queremos.

Por fim, inseriremos o `include_lowest` igual a `True` para incluirmos o limite mais baixo.

```
pd.value_counts(
    pd.cut(
        x = dados.Renda,
        bins = 17,
        include_lowest = True
    )
)
```

(-200.001, 11764.706]	75594
(11764.706, 23529.412]	1022
(23529.412, 35294.118]	169
(35294.118, 47058.824]	19
(47058.824, 58823.529]	16
(94117.647, 105882.353]	6
(58823.529, 70588.235]	5
(70588.235, 82352.941]	4
(188235.294, 200000.0]	3
(117647.059, 129411.765]	1
(82352.941, 94117,647]	1
(105882.353, 117647.059]	0
(176470.588, 188235.294]	0
(129411.765, 141176.471]	0
(141176.471, 152941.176]	0
(152941.765, 164705.882]	0
(164705.882, 176470.588]	0

Neste resultado, veremos as 17 classes.

Notaremos que a classe com `(188235.294, 200000.0]` deveria estar em último lugar, já que possui o valor máximo.

Isso acontece porque a função `value_counts()` faz uma ordenação pelos valores da outra coluna, a qual está organizada do maior ao menor número

Como queremos exibir a ordem correta das classes, o parâmetro `sort` deverá ser igual a `False` .

```
pd.value_counts(
    pd.cut(
        x = dados.Renda,
        bins = 17,
        include_lowest = True
    ),
    sort = False
)
```

(-200.001, 11764.706]	75594
(11764.706, 23529.412]	1022
(23529.412, 35294.118]	169
(35294.118, 47058.824]	19
(47058.824, 58823.529]	16
(58823.529, 70588.235]	5
(70588.235, 82352.941]	4
(82352.941, 94117,647]	1
(94117.647, 105882.353]	6
(105882.353, 117647.059]	0
(117647.059, 129411.765]	1
(129411.765, 141176.471]	0
(141176.471, 152941.176]	0
(152941.765, 164705.882]	0
(164705.882, 176470.588]	0
(176470.588, 188235.294]	0
(188235.294, 200000.0]	3

Desta forma, veremos a organização como queríamos. Para organizarmos do mesmo jeito que fizemos antes, colocaremos o código dentro de `frequ`

```
frequencia = pd.value_counts(
    pd.cut(
        x = dados.Renda,
        bins = 17,
        include_lowest = True
    ),
    sort = False
)
```

Com a execução, criaremos a tabela de frequências sem exibi-la por enquanto.

Na célula seguinte, também usaremos o `percentual` da mesma maneira vista anteriormente, sem esquecermos do `normalize` igual a `True` .

Executaremos e depois escreveremos `percentual` ao final do bloco para visualizarmos a tabela de fato.

```
percentual = pd.value_counts(
    pd.cut(
        x = dados.Renda,
        bins = 17,
        include_lowest = True
    ),
    sort = False,
    normalize = True
)
percentual
```

(-200.001, 11764.706]	0.983784
(11764.706, 23529.412]	0.013300
(23529.412, 35294.118]	0.002199
(35294.118, 47058.824]	0.000247
(47058.824, 58823.529]	0.000208
(58823.529, 70588.235]	0.000065
(70588.235, 82352.941]	0.000052
(82352.941, 94117,647]	0.000013
(94117.647, 105882.353]	0.000078
(105882.353, 117647.059]	0.000000
(117647.059, 129411.765]	0.000013
(129411.765, 141176.471]	0.000000
(141176.471, 152941.176]	0.000000
(152941.765, 164705.882]	0.000000
(164705.882, 176470.588]	0.000000
(176470.588, 188235.294]	0.000000
(188235.294, 200000.0]	0.000039

Seguindo a mesma metodologia aplicada anteriormente, na célula seguinte criaremos a variável `dist_freq_quantitativas_amplitude_fixa` sendo

Por fim, chamaremos a variável, executaremos a célula e veremos o resultado.

```
dist_freq_quantitativas_amplitude_fixa = pd.DataFrame(
    {'Frequência': frequencia, 'Porcentagem (%)': percentual}
)
dist_freq_quantitativas_amplitude_fixa
```

	Frequência	Porcentagem (%)
(-200.001, 11764.706]	7594	0.983784
(11764.706, 23529.412]	1022	0.013300
(23529.412, 35294.118]	169	0.002199
(35294.118, 47058.824]	19	0.000247
(47058.824, 58823.529]	16	0.000208
(58823.529, 70588.235]	5	0.000065
(70588.235, 82352.941]	4	0.000052
(82352.941, 94117,647]	1	0.000013
(94117.647, 105882.353]	6	0.000078
(105882.353, 117647.059]	0	0.000000
(117647.059, 129411.765]	1	0.000013
(129411.765, 141176.471]	0	0.000000
(141176.471, 152941.176]	0	0.000000
(152941.765, 164705.882]	0	0.000000
(164705.882, 176470.588]	0	0.000000
(176470.588, 188235.294]	0	0.000000
(188235.294, 200000.0]	3	0.000039

Feito isso, veremos a tabela bem organizada para ser apresentada.

Caso queiramos, poderemos aplicar outros macetes interessantes, títulos e labels.

A seguir, veremos como visualizar as Distribuições de Frequência em **forma gráfica** por meio dos **Histogramas**.